# Data Structures and Java

## Assignment 2 – Read/Write Input/Output (IO)

### Assignment objectives:

In this assignment, you may use your knowledge of the following concepts:

1. Reading from console and files (you can use either BufferedReader or Scanner)
2. Writing to console and files (use BufferedWriter to write to file and you can use System.out (a PrintStream) or a PrintWriter to write out to console))
3. Loops (do-while) and condition flows (if-else)
4. Testing for String and number equality
5. Calling methods and sending arguments
6. Try-catch-finally

In the template provided, there are methods to: 1) request a play by play report for a single game from the NHL.com and 2) extract shot data therefrom. The extracted data, which is a line of comma separated values will be used for this assignment. The report used is for a game between the Tampa Bay Lightning and the Chicago Blackhawks and is located here: <http://www.nhl.com/scores/htmlreports/20132014/PL020026.HTM>

In the play by play report there are details for every event in the game that the NHL tracks. In this assignment, we’ll be working with shot events.

### Tasks

The examples and output below reference the 3rd and 5th shots of this game. Your solution should work for at least these two shots.

The HTML for the 3rd shot is as follows:

<td class=" + bborder">CHI ONGOAL - #20 SAAD, Wrist, Off. Zone, 53 ft.</td>

The HTML for the 5th shot is as follows:

<td class=" + bborder">CHI ONGOAL - #2 KEITH, Slap, Off. Zone, 61 ft.</td>

#### Task 1

Create a new project in NetBeans with a class called A2FirstNameLastName.java. Integrate the code provided in the template.

#### Task 2

The program should do the following:

1. Print a welcome message: *“Welcome to the NHL play by play event scraper. The system will get events for the specified game for the specified event type.*”
2. Ask the user for input: “Please enter a valid game number: “. E.g. 20105 which is the 105th regular season game for the current season. Read the input and store it in a variable called int gameNumber.
3. If the user enters a string or any special character(basically not a valid number)

Print “Invalid game number. Please try again:”

Else if the user enters a game number less than 20001 or greater than 21230

Print “Invalid game number. Please try again:”

Go back to step 2

Else if the user enters “exit”

Print “Bye”

Exit the program

Else (it is a valid game number)

Go to step 4

1. Ask the user for input: “Please enter a valid event type: “ (i.e. the only valid event types is SHOT). Read the input and store it in a variable called String eventType.
2. If the user enters an invalid event type

Print “Invalid event type. Please try again:”

Go back to step 4

Else if the user enters “exit”

Print “Bye”

Exit the program

Else (it is a valid event type)

Go to step 6

1. Ask the user for which event in the game to extract: “Please enter the nth " + eventType + " you would like:”
2. If the user enters a non-positive integer

Print “Invalid event number. Please try again:”

Go back to step 6

Else if the user enters “exit”

Print “Bye”

Exit the program

Else (it is a valid event type)

Go to step 8

1. Call the method getNthEventByType() to get the nth shot from the HTML report
2. Call the method getShotDataFromEventHTML() to extract the shot data in a comma separated String
3. We will use a method called writeRecordToFile(int gameNumber, String eventType, String record) to write our records to file. This method will accept three inputs and will write/append to record to a file called <game number>\_<event\_type>.csv. This method returns and int. In main(), make sure to save the return value in and int called writeStatus.
4. After control has returned from writeRecordToFile(), report the value of the writeStatus as follows: “The write status is: ” + writeStatus
5. Write code for another method called printRecordsFromFile(int gameNumber, String eventType) that reads from <game number>\_<event\_type>.csv and prints the content to screen. This method has two inputs: 1) game number and 2) event type. This method should also return an int which should be stored in a variable called readStatus in main(). This method should be called from main() after writeRecordToFile() has been called.
6. After control has returned from printRecordsFromFile(int gameNumber, String eventType), report the value of readStatus as follows: “The read status is: “ + readStatus
7. Return to step 2.

**writeRecordToFile(int gameNumber, String eventType, String record)**

The new method called writeRecordToFile accepts three inputs, the game number, the event type and the record, and writes the record out to a file called <game number>\_<event\_type>.csv in append mode (make sure that you are NOT overwriting the file).

We want to keep track of whether the write operation was successful using an int called success. In the try{} block, if there are no errors, change the value of success to 1. If there are IOExceptions and you fall into the catch{} block, print an appropriate error message, and change the value of success to 0. Before ending the method, return this int success variable back to the main() method.

Some parts of the code for this new method is given below. You have to complete it.

**public static int writeRecordToFile(int gameNumber, String eventType, String record) throws IOException {**

// define the following variables

BufferedWriter outputBuff = null;

int success = -1;

return success;

}

**printRecordsFromFile(int gameNumber, String eventType)**

**public static int printRecordsFromFile (int gameNumber, String eventType) throws IOException {**

// write code to read line by line from file and print to screen.

// also create a success variable to indicate the status of the file IO operation

int success = -1;

return success;

}

### Deliverables

Please zip your entire project folder containing the **src** sub-folder and the **nbproject** subfolder (the others are not necessary). Name the zip file **A2FirstNameLastName.zip** and submit to Canvas.

### Sample Output

